AFRL-RI-RS-TR-2010-204

POLICY COMPLIANCE OF QUERIES FOR PRIVATE INFORMATION RETRIEVAL

Massachusetts Institute of Technology

November 2010

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2010-204 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
GENNADY R. STASKEVICH	JULIE BRICHACEK, Chief
Work Unit Manager	Information Systems Division
	Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* November 2010	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* June 2007 – May 2010

4. TITLE AND SUBTITLE POLICY COMPLIANCE OF QUERIES FOR PRIVATE INFORMATION RETRIEVAL	**5a. CONTRACT NUMBER** FA8750-07-2-0031
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER N/A
6. AUTHOR(S) Lalana Kagal	**5d. PROJECT NUMBER** NICE
	5e. TASK NUMBER 00
	5f. WORK UNIT NUMBER 12
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Massachusetts Institute of Technology, Decentralized Information Group (DIG), 32 Vassar Street, Cambridge MA 02139	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RISC, 525 Brooks Road, Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RISC
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2010-204

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2010-6173
Date Cleared: 19 November 2010.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The use of Private Information Retrieval (PIR) techniques enable clients to retrieve items from cooperating databases without revealing either the queries or the information being retrieved. In order to prevent clients from accessing information that they are not authorized to access, it must be possible to prove that the queries being posed are compliant with a set of privacy policies previously agreed upon by the clients and database owners. Efforts to address privacy in these situations have been dominated by techniques that assume that most clients are malicious and focus on helping database owners restrict access to data. With the current push towards need-to-share, we suggest alternative approaches such as the application of accountability mechanisms. These mechanisms include the use of formalisms that can express realistic data-use policies, automated reasoning engines that can interpret those policies, automatically determining whether particular queries are policy-compliant, and justifications to enable users to understand the compliance decision and the policies.

15. SUBJECT TERMS
Semantic Accountability, Information Sharing, Data Privacy, Data Transparency

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** GENNADY R. STASKEVICH
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	46	**19b. TELEPHONE NUMBER (*Include area code*)** N/A

Table of Contents

List of Figures

Abstract

The use of Private Information Retrieval (PIR) techniques enable clients to retrieve items from cooperating databases without revealing either the queries or the information being retrieved. In order to prevent clients from accessing information that they are not authorized to access, it must be possible to prove that the queries being posed are compliant with a set of privacy policies previously agreed upon by the clients and database owners. Efforts to address privacy in these situations have been dominated by techniques that assume that most clients are malicious and focus on helping database owners restrict access to data. With the current push towards need-to-share, we suggest alternative approaches such as the application of accountability mechanisms. These mechanisms include the use of formalisms that can express realistic data-use policies, automated reasoning engines that can interpret those policies, automatically determining whether particular queries are policy-compliant, and justifications to enable users to understand the compliance decision and the policies.

1. Summary

We have designed an architecture that enables policy compliance of PIR queries based on policy-awareness and accountability principles. We have implemented a Semantic Web based policy language, and an associated reasoner. Its purpose is to: infer whether queries are compliant, generates justifications and handles private policies through a user interface that allows justifications to be explored. Also we have developed various techniques for integrating our architecture with different databases. Using a semantic policy language gives policies explicit semantics that allows all participants to unambiguously understand their meaning. The justifications generated by checking incoming requests against these policies help requesters formulate privacy-aware queries. Lastly, reasoning over event logs and justifications allows data owners to verify that their privacy policies are being correctly enforced.

2. Introduction

The use of Private Information Retrieval (PIR) techniques enable clients to retrieve items from cooperating databases without revealing either the queries or the information being retrieved. However, as both the queries and results are hidden from the database owners, it is in principle possible for clients to access information that they are not authorized to access. In order to prevent this, it must be possible to prove that the queries being posed are compliant with a set of privacy policies previously agreed upon by the clients and database owners. **Policy compliance** deals with proving that client queries conform to mandated policies and that leakage of sensitive information is not possible.

Efforts to address privacy in these situations have been dominated by strict access restriction and privacy-preserving algorithms such as anonymization, generalization, and perturbation. These techniques assume that most clients are malicious and focus on helping database owners restrict access to data. However, these techniques also discourage honest clients from accessing data they require. With the current push towards need-to-share, we suggest alternative approaches that help honest clients be honest as well as enforce privacy policies of database owners.

We have explored the application of accountability mechanisms as a means of protecting central information policy values such as privacy and fair-use of information. We view accountability combined with transparency and appropriate redress as a complementary process to strict access control [11]. Fundamental to this approach is the use of formalisms that can express realistic data-use policies, automated reasoning engines that can interpret those policies and automatically determine whether particular uses of data are policy-compliant.

Most compliance mechanisms are based on strict binary principles and provide a simple yes/no or permit/deny kind of answers. These type of responses often confuse the clients, leaving them unsure of what they are doing is wrong and why. Clients wanting to comply with security and privacy policies end up either being afraid to make queries for fear of doing something wrong or being angry that they are never able to get a query past the security mechanism to the data they require. Our objective is to use *policy-awareness* to enable clients to make the right queries by giving them enough information to do so. Consider a simple privacy policy, “You may not query for both zip-code and last name”. With normal access control mechanisms, a client who is asking for several fields including zip-code and last name from the database will be constantly rejected without knowing why. Similarly a client who initially asked for zip-codes and is now asking for first and last names will be rejected. Our policy tools will provide a reason for this rejection such as, “You've previously requested zip-code. Once one of zip-code and last name has been requested, the other cannot be retrieved”. This justification enables the client to reformulate his query so that it does respect the privacy policies. Further, policies derived from regulatory controls governing information sharing are often complicated and may need to draw on contradictory sources from different jurisdictions [3, 4]. Examination of explicit justifications can expose these contradictions and lead to redefinition of the appropriate policy.

Policy-awareness also supports extensive and machine-readable logs of events that have occurred in the system including requests for data and the results of those requests. Our tools allow database owners to run audits over these logs to check whether the enforcement mechanisms are fulfilling the intent of the policy. In the case of non-conformance, the justifications associated with the decision provide guidance on how the policy should be adjusted to meet their intended result. For example, if the log shows that Alex accessed certain records the database owner or an authorized client can ask why this was permitted. Our tools provide a detailed justification saying that Alex is a Federal Bureau of Investigation (FBI) employee, and he had authorization, thus he was permitted to do so. This approach enables the database owner to enforce specific rules that ensures that FBI agents should have access only to relevant data (e.g. person of interest) and no one more.

2.1 Why Semantic Web Technologies

The entities in private information retrieval systems require the flexibility to exchange information, queries, and results with some assurance that they share a common meaning. The lack of a common understanding of shared information opens up new security and privacy vulnerabilities. The Semantic Web is an enhancement of the World Wide Web that deals primarily with data instead of documents, which were the focal point of the World Wide Web (WWW). It enables Web resources to be annotated with machine understandable meta-data, allowing the automation of the retrieval and usage of these resources in their correct contexts. Semantic Web technologies include languages for defining ontologies and describing meta-data using these ontologies and tools for reasoning over these descriptions.

Several policy languages in the literature could be used for policies about PIR queries within a single security domain. A single domain typically has a uniform domain specific semantics, is uniformly managed, and typically was not designed with cross-domain sharing in-mind. Though there has been some research on the problem of multiple security domains, it usually involves prior negotiation between separate domains in advance of the sharing that prevents dynamic sharing or sharing of new unaccounted-for information. We propose the use of Semantic Web technologies for encoding policy descriptions that span across multiple domains. It designed to facilitate the exchange of information between homogeneous and heterogeneous systems with some assurance that they share a common meaning. The Semantic Web is composed of multiple standards such as Resource Description Framework (RDF), RDF Schema (RDFS), and the Web Ontology Language (OWL) that enable the exchange of data, including policies and credentials, to be annotated with machine- understandable meta-data, and used in correct contexts.

The policy language AIR is based on Semantic Web technologies allowing it to be domain independent. The AIR framework enables the sharing of data across domains that adhere to the policies even though they have their own native schemas or data models. For example, it is possible to define a policy in AIR that describes the characteristics of users, the kinds of data that can be exchanged, and the conditions under which the exchange is possible between two government agencies. Therefore it is possible to share information between agencies while having different internal data models, role descriptions, user attributes, etc. We use Semantic Web technologies not only to model policies but also for our justifications and logs. Having common semantics for this security information enables our tools to be used in any system that understands basic Semantic Web technologies and allows participants from different domains to effectively use and reason over each others policies, logs and justifications.

2.2 Challenges

In this project, we have addressed three main challenges --- understanding the policy structure, expressing these policies in a suitable language and using appropriate reasoning methods. Also we have designed and developed appropriate user interfaces for creating policies as well as for viewing the reasoning results.

2.3 Policy Structure

Policies associated with PIR queries need to understand the query language as well as the specific structure of the query. Most PIR queries deal with retrieving values from tables and use other values to filter search results. For example, the query "Select the names of people who reside in New England" retrieves **names** and filters based on people who reside in New England. It must be possible to write different kinds of policies that are able to address both these functions. An example policy is, "The user may not query the names and addresses of people living in New England". It should also be possible to customize policies that are database specific without requiring significant modification to the policy compliance framework. A combination of individually compliant queries can still violate a policy. Therefore, it is imperative that the history of client's queries is taken into consideration. In short, the policy structure should be based on (i) the query language, (ii) the database schema, and (iii) the query history.

2.4 Policy Language

The policies need to be represented in a machine-understandable policy language so that we can reason over them and decide about compliance. However, there are several requirements that a suitable policy language has to meet.

The policy language should be open and not be restricted to any kind of domain specific data. Our policy compliance framework is domain agnostic and is capable of enforcing rules across heterogeneous databases. For example, data can include: biometric information, health records, student records and immigration data and capture different conditions over this data. For example, the language should support policies such as "access to Social Security Number (SSN) numbers for people born before 1920 is permitted", "my health information cannot be used to contact me regarding experimental drugs", "access to US residents for census purposes is permitted".

Another challenge is that policies are usually expressed at a higher level and are not always directly translatable into terms of the rows or columns within a database table. Typically policies are not as specific as "Person X is not allowed to access these fields in the database table if entity is a minor: home-email, work- email, home-phone, work-phone, home-address and work-address". This is an example of an abstract policy "access to contact information for minors is not permitted" where contact information is an abstract term consisting of address, phone numbers, email address, etc. that are columns from the table and minors is a set of rows defined by a constraint on age. The policy language should be able to express *contact information* and *minors* and define access rights over them.

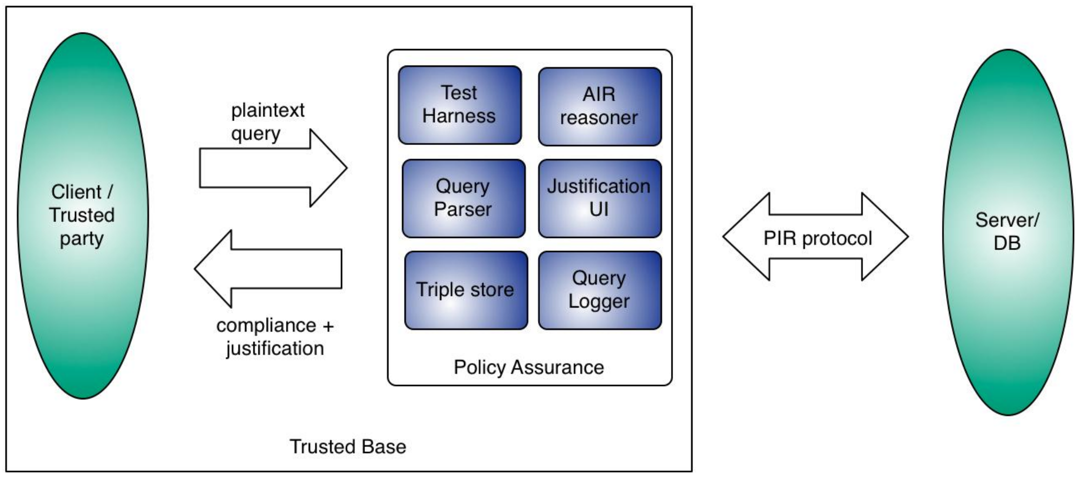


Figure 1. Architecture

Our research prototype consists of several different components as illustrated in Figure 1. A client sends plain text queries to the policy compliance prototype, which sends back the compliance result as well as the justification. Our prototype consists of a test harness, query parser, query logger, policy language and reasoner, an authoring tool to create policies, an interface to view the reasoning and justification results, and a database of external Semantic Web data.

Our project has a suitable policy language that is domain and data independent; it has the capability to support high level policies, and most importantly is capable of interpreting and reasoning over the query structure at runtime.

2.5 User Interfaces

Another requirement of policy compliance is having a set of suitable interfaces that support policy development and have the capability to retrieve, and review policy results. Policy authoring is known to be a difficult problem; hence, we defined a set of template policies based on database schema that users are able to customize and personalize as needed.

It is our belief that a mere identification of whether a query is or is not compliant is insufficient and that a reason or a justification for every compliance/non-compliance result should be provided. This justification will help end users ensure that the policy assurance process works as it is supposed to and in the case of incompliant queries to understand what it is that is not permitted. This will help users conform to the policy because they will better understand the different restrictions on the queries

Justifications produced by reasoners are usually expressed as proof-trees; often users find these justifications confusing or incomprehensible. We have developed a graphical interface that extracts relevant data from these proofs and allows the justification to be interactively explored.

3. Methods, Assumptions, Procedures

Our research prototype consists of several different components as illustrated in Figure 1, namely a test harness, query parser, query logger, policy language, reasoner, an authoring tool to create policies, an interface to view the reasoning and justification results, and a Semantic Web database also known as a triplestore. The client sends plain text queries to the prototype, which is within its trusted base, and the tool sends back the compliance result as well as the justification. The policy compliance process is separated from the PIR protocol that the client performs with the database server.

3.1 AIR Language and Reasoner

AIR (**A**ccountability **I**n **R**DF) is a Semantic Web-based rule language. It provides a version of production rules that can exist on separate Web servers but can be linked together. They are known as ***Linked Rules***, a key enabler for modular development of rules [6]. This modularity and nested activation reflects how rules are expressed in laws, security policies, business rules and work-flows. AIR is represented in TURTLE a subset of Notation 3 (N3) [1]; it is an abbreviated human-readable representation of the RDF data model. AIR supports a rich set of functions for selectively accessing trusted content from the Web as well as cryptographic, string, and math operations. Furthermore, AIR supports contextually scoped reasoning, i.e. rule conditions can be scoped to be satisfied against different datasets or against deductions from different rules and datasets.

Though, the initial development of AIR policy language preceded this project; we have made significant contribution to the AIR language. We developed the formal semantics of AIR [8] and defined the notion of ***Linked Rules***, and developed a methodology for customizing AIR policies to specific databases. Also, we developed semantically aware policies based on external data sources and modified how justifications were generated in order to better represent the AIR reasoning process.

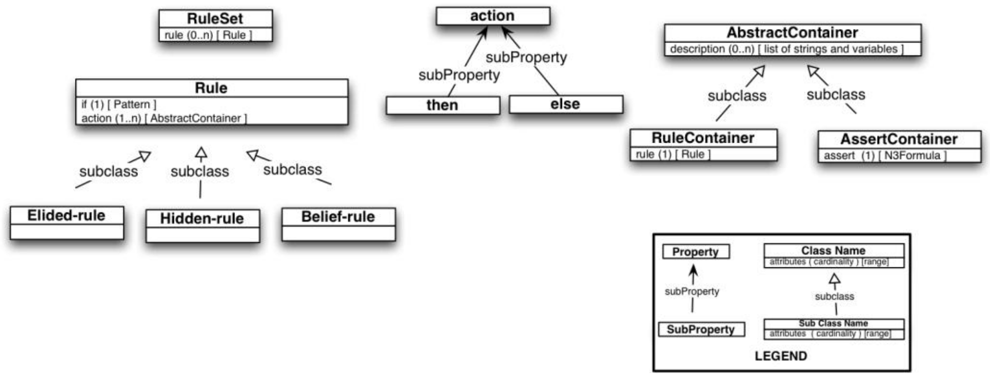


Figure 2. Air Rule Ontology

3.1.1 Overview of AIR Language

As illustrated in Figure 2, AIR rules are defined using the following properties: ***air:if***, ***air:then***, ***air:else***, ***air:description***, ***air:rule*** and ***air:assert***. Every rule is named with a Uniform Resource Identifier (URI), and they are grouped into ***air:RuleSets*** or nested under other rules. This nesting can happen either under the ***air:then*** property or the ***air:else*** property. The rules nested directly under the ***RuleSet*** are referred to as the top rules of the ruleset. A chain of rules is defined as a sequence of rules, such that every rule, barring the first in the chain, is nested under either the ***then*** or the ***else*** of the preceding rule.

There are three distinct types of rules in AIR --- ***air:Belief-rule***, ***air:Hidden-rule*** and ***air:Elided-rule***. All rules are, by default, ***Belief-rules***. The descriptions and conditions of ***Belief-rules*** contribute to the overall justification. In contrast, ***Hidden-rules*** and ***Elided-rules*** are used to modify the default justification. (Please refer to Section 4.1.3 for more information about justification generation and modification).

The conditions of a rule are defined as graph patterns that are matched against RDF graphs, similar to the Basic Graph Pattern (BGP) of SPARQL queries[1]. If the condition matches the current state of the world, defined as the facts known or inferred to be true, then all the actions under ***then*** are fired, otherwise all the actions under ***else*** are fired. The condition matches the current state if there is a subgraph of known facts that matches the graph pattern. This sub-graph is referred to as the matched graph.

Existentially quantified variables may be declared within graph patterns by using the **@forSome** directive. Any universally quantified variables, quantified using **@forAll**, and are declared outside of the rule. The scope of existentially quantified variables is the graph pattern in which it is declared, whereas that of a universally quantified variable is any chain of nested rules.

Nesting of rules can be used to specify conditions where some graph pattern must match the current state and others should not match the current state. The actions under ***then*** and ***else*** (together referred to as actions) are defined by an assertion pattern using the ***air:assert*** property, or a rule reference, using the ***air:rule*** property. All actions may be annotated with the natural-language description of the rule or action through the use of the ***air:description*** property. When the action is executed, the variables in an assertion pattern are substituted by their bindings and the pattern is asserted. If a rule reference is defined instead, an instance of that rule, created by substituting the variable bindings acquired so far, is activated. The variables in any ***air:description*** property are also instantiated, and the description is maintained by the reasoner.

3.1.2 Linked Rules

Most rules, whether laws, security policies, business rules, or workflow plans, are rarely defined by a single entity or exist in a single document. They usually comprise of several rules that are defined and maintained by different entities. For example, a University policy depends on concepts and rules defined by the HR (Human Resource) department, the different schools, as well as the labs within the schools. Additionally, rules often reference other rules, including those of other organizations. As an example, a hospital might want to know with whom a pharmaceutical company might share patient records it provides and might want to reason over the latter's rules before transferring patient information. AIR models this rule re-use and linkage by (i) enabling rules to be uniquely identified by URIs such that they can be spatially dispersed but combined during reasoning, (ii) allowing rules to be developed modularly using existing rules, and (iii) allowing rules to be executed from within other rules and their results to be queried. This conformance of AIR rules to Linked Data principles forms the basis of ***Linked Rules*** that provide a more natural way to think about and model real world rules, laws and policies on the Web.

[1] http://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns

3.1.3 Justification Generation and Representation

Upon finalizing its reasoning results, the AIR reasoner produces a justification that contains sufficient information to understand the decisions and actions made by the reasoner and to debug the rules, if needed. We have developed a justification ontology based on basic Proof Markup Language (PML) concepts [9]. The AIR operational semantics also define how this justification is generated by the reasoner. As AIR justifications themselves expressed as N3 data, they can be consumed by other reasoners, including AIR to evaluate the *quality* and *trustworthiness* of the results based on the rules and data sources used. Figure 3, the PML concepts are enclosed in solid boxes and ovals, sub-concepts and sub-properties are depicted in the boxes adjacent to the concept and property labels, respectively.

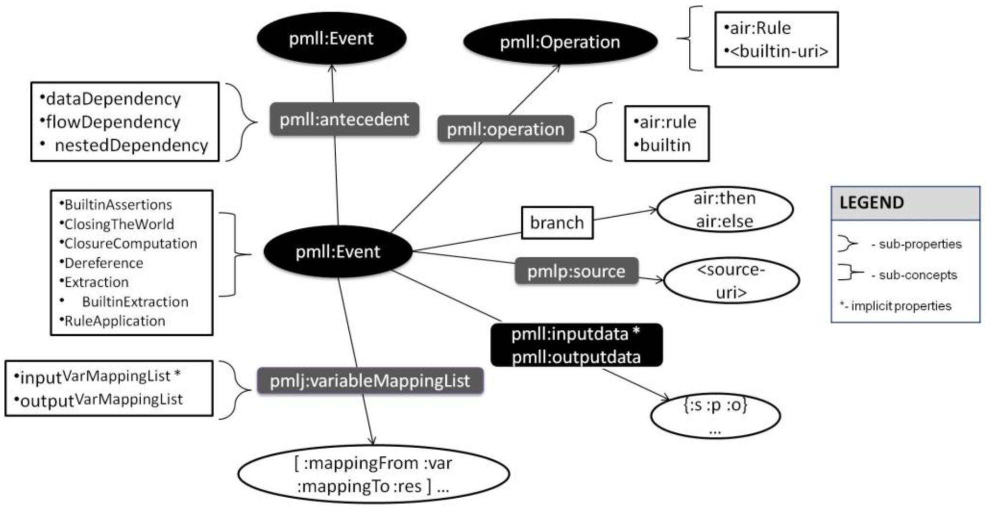


Figure 3. AIR Justification Ontology

3.1.3.1 Justification Ontology

The AIR justification ontology extends certain PML [9] concepts as shown in Figure 3. PML [12] is an Interlingua used to describe the justifications as a sequence of information manipulations that generated an answer. A PML proof can represent many kinds of information manipulations - formal logic derivations, natural deduction derivations, information retrieval operations, natural language processing, or policy decision derivations. PML-Lite[2], an event-based ontology, is a simpler vocabulary that captures many concepts of the PML 2.0 ontology [13]. PML-Lite vocabulary can be conveniently used for representing the AIR reasoning steps. The AIR justification ontology consists of three main classes --- ***pmll:Event***, ***pmlj:Mapping*** and ***pmll:Operation***. The ***pmll:Events*** may be categorized into ***airj:BuiltinAssertion***, ***airj:ClosingTheWorld***, ***airj:ClosureComputation***, ***airj:Dereference***, ***airj:Extraction***, and ***airj:RuleApplication*** events depending on the operation performed. These steps have data & control flow dependencies over one another: represented by: ***dataDependency***, ***nestedDependency*** and ***flowDependency***.

The AIR reasoner computes the closure of input facts with respect to a given AIR ruleset. An ***airj:ClosureComputation*** event captures this operation. The events of all other types are all part of some ***airj:ClosureComputation*** event.

[2] http://tw.rpi.edu/proj/tami/PML-Lite

The input to an AIR ruleset may be contained in a set of N3 or RDF documents. These documents, on the web or on a local machine, are dereferenced to retrieve their RDF representation. This step is encoded as an ***airj:Dereference*** event.

Apart from the input triples, there are RDF triples that are tautologically true. Some of these are built-in assertions. In practice, built-in triples are created dynamically. However, we abstract this process and represent it through ***airj:BuiltinAssertion*** and ***airj:BuiltinExtraction*** events. The output of an ***airj:BuiltinAssertion*** event is assumed to be the graph that contains all the true built-in assertions (potentially unbounded in number) for the built-in function specified by the ***airj:builtin*** property. The assertions needed to match rule conditions are then extracted from this output in an ***airj:BuiltinExtraction*** event. The ***airj:Extraction*** event from which ***airj:BuiltinExtraction*** is derived, is used to encode the step where a subgraph is obtained from a graph. The input to ***airj:BuiltinExtraction*** event is also left implicit, but may be determined by the ***airj:BuiltinAssertion*** event on which the extraction has an ***airj:dataDependency***.

The ***airj:ClosingTheWorld*** event refers to a step of reasoning where triples other than the ones in the input or those inferred so far, and rules other than those that are active, are believed not to be true. An ***airj:ClosingTheWorld*** event has data and/or flow control dependencies on all prior ***airj:RuleApplication*** events. These dependencies are represented through ***airj:dataDependency*** and ***airj:flowDependency*** properties.

The ***airj:RuleApplication*** event represents a rule firing event. It is linked to the rule that fired with the air:rule property. When a nested rule is activated, the known variable bindings are passed from the parent rule. The ***airj:RuleApplication*** event for a nested rule has a special flow dependency, referred to as an ***airj:nestedDependency***, on the ***airj:RuleApplication*** event where the parent rule fired. The input variable bindings for the nested rule are implicit and are the same as the output variable bindings of the parent.

When the condition of the rule is satisfied, variable bindings may be acquired, and the ***air:then*** actions are effected. Otherwise, the ***air:else*** actions are effected. Which actions are effected is declared using the ***air:branch*** property. The variable bindings are acquired upon pattern matching of the rule condition against the fact base, and declared using the ***airj:outputVariableMappingList*** property. Any triples asserted when the rule fires are declared using the ***airj:outputdata*** property and the event is annotated with natural-language description specified by the ***air:description*** property. The ***airj:RuleApplication*** events may also have data dependencies on other events. The condition can be satisfied by triples from more than one input log, or by triples asserted in some of the prior ***airj:RuleApplication*** events. These events also have flow control dependencies on prior ***airj:ClosingTheWorld*** events.

We may determine the order in which rules were applied and what data was used through manual or an automated tracing of the output data and rule dependencies from each ***airj:BuiltinExtraction***, ***airj:Dereference***, or ***airj:RuleApplication*** event. This also allows for the construction of meaningful, human-readable justification traces and interfaces [5].

3.1.3.2 Justification Generation

AIR supports justification generation for every action taken by the reasoner. When an action is taken (either a graph pattern asserted or a rule activated), the action is annotated with the identity of the rule and with either the matched sub-graph or a list of components known to be true under the closed world assumption. The operational semantics define how the reasoner generates the justification. The default justification for an AIR conclusion is constructed by taking the union of annotations for the conclusion and the rule in which the conclusion was asserted.

Though knowing the rules and facts from which a conclusion is derived is useful, it does not describe what the rule was attempting to do. In order to provide natural-language explanations, we allow ***air:descriptions*** to be added to ***air:actions***. These descriptions are English sentences and can contain variable values. The ***air:description*** property is a list instance, where list items are enclosed in brackets and separated by commas. Each list item can either be a string enclosed in quotes or a quantified URI variable. During the reasoning process, each variable is replaced by its current value and inserted into the description string.

Sometimes, the default justification can be unwieldy or very revealing, and needs to be modified to hide trivial or sensitive information. AIR provides mechanisms to *declaratively modify justifications* generated by default. Rules in AIR may be declared to be ***air:Hidden-rules*** or ***air:Elided-rules*** to suppress justifications for certain actions. The detailed justifications for actions executed when an ***Elided-rule*** fires are suppressed, and only the natural-language description is provided. The justification for actions executed when a ***Hidden-rule*** or its descendants fire are suppressed completely. This flexibility to control the level of details, at a rule-based granularity, helps the rule writers to adjust the justification so that sensitive information is not revealed and so explanations are not overly verbose. Nesting of rules can be used advantageously to split a rule's conditions across multiple rules when parts of the graph pattern refer to sensitive (or insignificant) information, such that rules with sensitive conditions may be elided or hidden.

3.1.4 AIR Semantics

The procedural semantics of AIR describe how AIR rules fire and how inferences are made. The AIR reasoner applies forward chaining reasoning to compute the closure of AIR rules over the input data. When AIR rules fire, their actions, substituted with known variable bindings, are effected. As a result, new rules may be added to the rule base, i.e. new rules may be activated, and/or facts may be deduced. Initially the rule base contains all top rules in the ruleset, and the fact base is made of the input facts. The rules in the rule base are said to be active. The active rules whose conditions match the current state of the world (fact base) are referred to as successful rules, whereas those active rules whose conditions have no match are called failed rules.

AIR reasoning is performed in stages. In any given stage, the successful rules are given priority over failed rules and their then-actions are effected before failed rules fire. When all successful rules have fired the world is temporarily closed and the else-actions of all the failed rules are fired simultaneously with the belief that the conditions of all the failed rules are false. AIR reasoning enters the next stage once the failed rules have all fired. The AIR closure is computed in a finite number of stages [7].

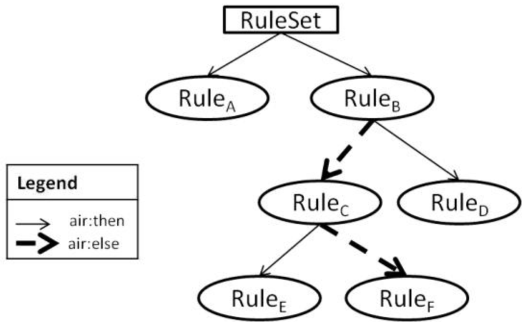


Figure 4. Example of AIR Rule Nesting

In order to illustrate rule nesting, we consider an arbitrary nesting shown in Figure 4. Initially, i.e. before entering stage 1, only Rule$_A$ and Rule$_B$ are in the rule base. Then, if the condition of Rule$_B$ is satisfied, Rule$_D$ would be active in stage 1. However, if in stage 1 Rule$_B$ doesn't succeed, then Rule$_C$ will be added to the rule base after the world is closed for stage 1, and will be active from stage 2. Now, in stage 2, if Rule$_C$ succeeds then Rule$_E$ will become active in stage 2. Otherwise Rule$_E$ will be active from stage 3. Note that if Rule$_B$ succeeds in later stages, say stage 3, then Rule$_D$ will also be active (in addition to Rule$_C$) from stage 3 onwards.

The declarative semantics of an AIR program are defined through translation to stratified Logic Programs [10]. The AIR-closure computation is polynomially-complete in data-complexity and exponentially-complete in program-complexity, where data complexity is the complexity of computing the closure when the program is fixed and the facts are input and program complexity is the complexity when the facts are fixed and program is input [7].

Nesting introduces an ordering of rules that can be leveraged to encode fairly expressive Logic Programs (LP) in AIR, such as Positive LP and a special class of stratified LP - Positively Stratified Negatively Hierarchical LP [7]. Furthermore, SPARQL SELECT and CONSTRUCT queries may be encoded in AIR and executed by the AIR reasoner [7].

3.1.5 Database Customization

As policies require the database structure in order to restrict/permit access to data in the database, this structure needs to be provided as input to the policy compliance tools. By using a domain independent policy language, our tools are able to support any database structure as long as it is defined in a Semantic Web language such as RDFS or OWL.

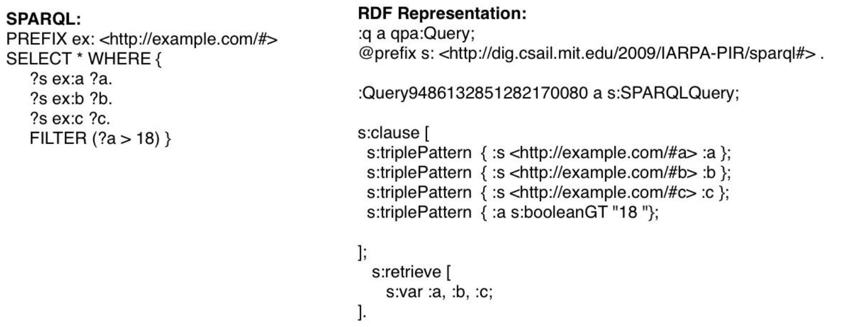
Even though using the database structure is a good starting point, we found that most policies are more abstract and at a higher level policies than the column or row names of a table. For example, a policy might state Access to contact information of minors is not permitted. In this case, *contact information* is not the name of any specific database column but a collection of several columns such as **email address**, **mailing address**, **telephone number**, and fax **number**. Another policy example is, “The user may not filter based on Midwest”. In this case, we don't even know what columns the policy is talking about we only know that the range of those fields must not be a value that is in the Midwest. We provide two methods of supporting these abstract policies --- (i) grouping attributes and (ii) defining ontological relationships between columns. For example, *contact details* is defined as a group containing several columns including **emai**l, **address**, **telephone**, **fax number** and **office add** and policies will refer to *contact details* concept. As an example of using ontological relationships consider *HealthInfoData*. It can be defined as a class and **CurrentSymptoms**, **Prescriptions,** and **Illnesses** are subclasses of *HealthInfoData* that can have their own properties. Policies dealing with health information would specify access to a particular subclass or property of the *HealthInfo* class hierarchy.

3.1.6 Semantically Aware AIR Policies

Along with database meta-data such as the schema and abstract terms, database owners can also provide additional information such as references to online Semantic Web data that can be used for better control and expressivity in policies. For example, consider the policy "The user may not query specifically for people living in New England". In this case, the policy tools need to be told how to infer the meaning of "New England" and what the "lives in" concept means. "Lives-in" can be defined as an abstract concept and a grouping of columns of the database namely city, state, and zip-code. However, "New England" cannot be defined in terms of the database schema and is defined in terms of existing online data such as the US census information or Semantic Web data sources such as DBPedia[3]. This can be done either using AIR or RDF. Policies can then be written in terms of "New England" and while checking policy compliance, our tools will automatically execute the AIR or RDF definition of "New England".

3.2 Query Parser

Earlier in this project, we concentrated on supporting SPARQL Query Language for RDF in our policies as SPARQL is the query language for Semantic Web data. This allowed for easier integration with the rest of the tools and was a logical first step. SPARQL, unfortunately, is not in RDF and so we had to develop tools to translate SPARQL queries into RDF to be used by our policy compliance prototype. Our first attempt at SPARQL translation leads to a detailed ontology in RDF that captured most of the semantics of SPARQL. Though this was useful research, it lead to lengthy and complex policies. We realized that we could not continue with this translation, so we tried to come up with a simplified ontology. This ontology actually flattened the earlier ontology causing us to lose most of the semantics of SPARQL. But we found that it captured the essential parts of the query structure and also reduced the size and complexity of our policies. The ontology was the smallest ontology we could come up with that maintained the components of the query that we require for reasoning. We host an online service (http://dig.csail.mit.edu/2009/policy-assurance/sparql2n3.py) that accepts SPARQL queries and returns the translated query in our simplified ontology. An example of a translated SPARQL query is illustrated in Figure 5.



SPARQL:

```
PREFIX ex: <http://example.com/#>
SELECT * WHERE {
    ?s ex:a ?a.
    ?s ex:b ?b.
    ?s ex:c ?c.
    FILTER (?a > 18) }
```

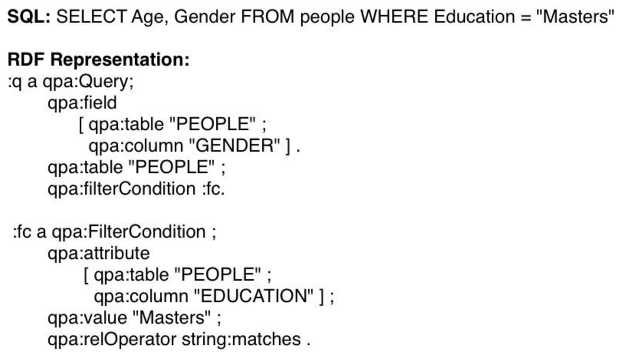
RDF Representation:

```
:q a qpa:Query;
@prefix s: <http://dig.csail.mit.edu/2009/IARPA-PIR/sparql#> .

:Query9486132851282170080 a s:SPARQLQuery;

s:clause [
  s:triplePattern  { :s <http://example.com/#a> :a };
  s:triplePattern  { :s <http://example.com/#b> :b };
  s:triplePattern  { :s <http://example.com/#c> :c };
  s:triplePattern  { :a s:booleanGT "18 "};

];
   s:retrieve [
      s:var :a, :b, :c;
].
```

Figure 5. SPARQL Translation Ontology

[3] http://dbpedia.org/About

During the last few months of the project, we started working on supporting SQL queries. After discussion with MIT Lincoln Labs (MIT-LL), we decided that it is more likely that real world applications of the PIR technology would use SQL rather than SPARQL. Our query parser now accepts SQL queries and converts them into RDF so that our tools can reason over them. Figure 6 provides an example of translating an SQL query into RDF.



SQL: SELECT Age, Gender FROM people WHERE Education = "Masters"

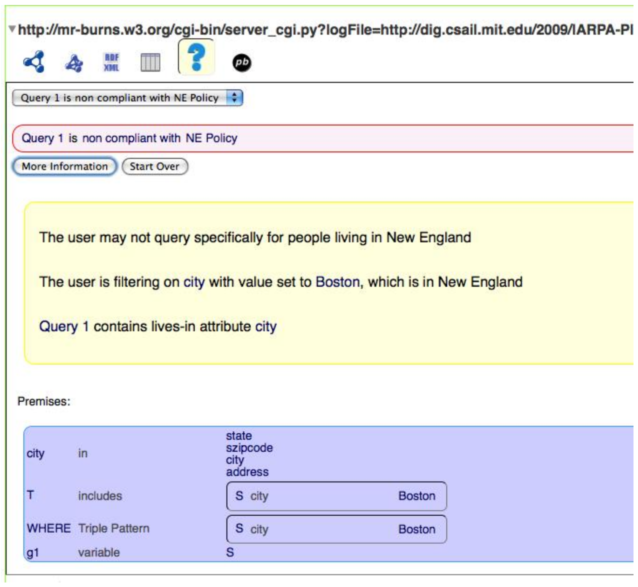
RDF Representation:

```
:q a qpa:Query;
      qpa:field
            [ qpa:table "PEOPLE" ;
              qpa:column "GENDER" ] .
      qpa:table "PEOPLE" ;
      qpa:filterCondition :fc.

 :fc a qpa:FilterCondition ;
      qpa:attribute
            [ qpa:table "PEOPLE" ;
              qpa:column "EDUCATION" ] ;
      qpa:value "Masters" ;
      qpa:relOperator string:matches .
```

Figure 6. Translation of an SQL query to RDF

3.3 Justification User Interface (UI)

The proofs generated by the AIR reasoner are in the form of proof trees. These proof trees are not easy for end users such as policy administrators to understand and also contain a lot of irrelevant information such as the variable bindings. These proof trees, however, do in fact contain all the information required for useful explanations, but the relevant information needs to be extracted and presented in an easy to understand manner. We developed a graphical justification user interface in Tabulator [2], a Semantic Web browser. This allows users to view the explanation provided by the reasoner in different ways: (i) in a simple rule language, N3, and (ii) in a graphical layout that highlights the result of the reasoning and allows the explanation to be explored. Please refer to Figure 7 for the graphical view of an AIR justification. The **More Information** button provides a way for the user to step through the proof starting from the most relevant information and working backwards to the top of the tree. The top bar contains the description of the rule that generated the assertion and the bar below contains premises of that rule.



http://mr-burns.w3.org/cgi-bin/server_cgi.py?logFile=http://dig.csail.mit.edu/2009/IARPA-Pl

Query 1 is non compliant with NE Policy

Query 1 is non compliant with NE Policy

More Information | Start Over

The user may not query specifically for people living in New England

The user is filtering on city with value set to Boston, which is in New England

Query 1 contains lives-in attribute city

Premises:

city	in	state szipcode city address
T	includes	S city Boston
WHERE	Triple Pattern	S city Boston
g1	variable	S

Figure 7. Justification Unser Interface

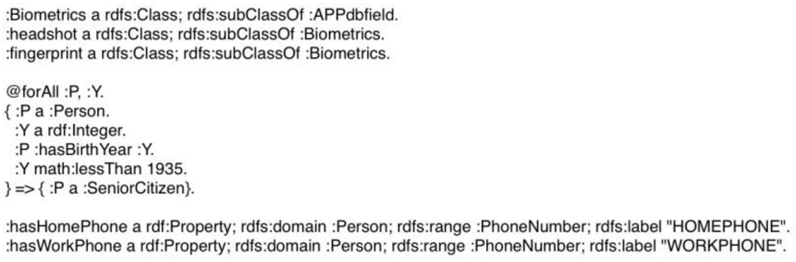
3.4 Semantic Web Database/Triple Store

Until recently the AIR reasoner had to load everything in memory before reasoning. This greatly reduced the amount of data it could handle. We modified the reasoner to support large datasets and external data sources by means of a family of built-in functions. Web documents can be accessed using the built-in function ***log:semantics***, and matching functionality is provided by the ***log:includes*** property. The ***log:semantics*** and ***log:includes*** properties allow rules to extract certain RDF data from Web documents making them useful in *trust management* as different Web documents may be trusted with assertions about certain data but not all data. Accessing only trusted RDF subgraphs from Web pages is useful in maintaining the quality of inference results. The ***log:notIncludes*** property is used to check if a graph is not included in another graph (with closed world semantics). Semantic Web databases also known as SPARQL endpoints can be queried from within AIR policies using ***sparql*** built-ins. By loading input data into a trusted SPARQL endpoint and only retrieving data that is required using the ***sparql*** built-ins, AIR is able to reason over much larger datasets. Other built-in functions that AIR supports includes those for cryptographic, math, string, list and time functions. ***:LEN math:notGreaterThan 6*** is an example graph pattern that uses the **math:notGreaterThan** built-in function. The subgraph of ***:LEN math:notGreaterThan 6*** matches if the value of **:LEN**, which is a variable, is found to be less than or equal to 6. Arguments to N-ary functions and built-ins are declared using an **rdf:List**.

4. Results and Discussion

4.1 Test Harness

The test harness is a very simple python script that is used by MIT-LL's test script to test our prototype. The harness allows the queries to be sent to the prototype and returns the result in the format required by MIT-LL. It parses the incoming SQL query and generates an RDF representation of the query. This query is then sent to the AIR reasoner. The result provided by the AIR reasoner is stored in a Web accessible location on the server so that it can be viewed remotely.



```
:Biometrics a rdfs:Class; rdfs:subClassOf :APPdbfield.
:headshot a rdfs:Class; rdfs:subClassOf :Biometrics.
:fingerprint a rdfs:Class; rdfs:subClassOf :Biometrics.

@forAll :P, :Y.
{ :P a :Person.
  :Y a rdf:Integer.
  :P :hasBirthYear :Y.
  :Y math:lessThan 1935.
} => { :P a :SeniorCitizen}.

:hasHomePhone a rdf:Property; rdfs:domain :Person; rdfs:range :PhoneNumber; rdfs:label "HOMEPHONE".
:hasWorkPhone a rdf:Property; rdfs:domain :Person; rdfs:range :PhoneNumber; rdfs:label "WORKPHONE".
```

Figure 8. Part of the database structure used for evaluation

4.2 Evaluation

Our prototype was evaluated by MIT-LL in April using the test harness. MIT-LL gave us eighteen policies of increasing complexity in English and provided thirteen users, each of which could have between 1 and 18 policies. We converted the policies into AIR and described the users using Friend Of A Friend (FOAF) ontology[4]. Each user was tested with approximately 100 queries making the total number of queries approximately 1300. Figure 8 shows portion of the database structure used for the evaluation. MIT-LL checked false positives and false negatives and calculated the policy compliance metric defined in Figure 9 to ensure that our tools meet the requirements. Though we did not receive detailed evaluation results, we were informed that our prototype not only met but exceeded the required value (80) for the metric.

$$\frac{N}{N_{correct} + 1.5*N_{fp} + 2*N_{fn}}$$

Figure 9. Evaluation metric

[4] http://www.foaf-project.org/

5. Conclusion

We have previously conducted significant research into the expression of policies and the automated compliance checking of these policies in various domains using Semantic Web standards and technologies. In this project, we leveraged our research to design and implement a set of proof-of-concept policy assurance tools for PIR queries. Figure10 illustrates the timeline of our achievements. Our major tasks included developing a methodology for converting privacy policies to query-based policy compliance rules, modifying our current policy language and reasoner to support these kinds of rules, representing as machine-understandable the set of policies and queries prepared by the test and evaluation team, and meeting the policy compliance metric. Our sub-tasks included developing a graphical user interface to semi-automatically define these policies, and capturing the semantics of SQL queries in Semantic Web languages for easier integration with our prototype. Along with developing a research prototype we also worked on the conceptual foundations of accountability as described in [11].

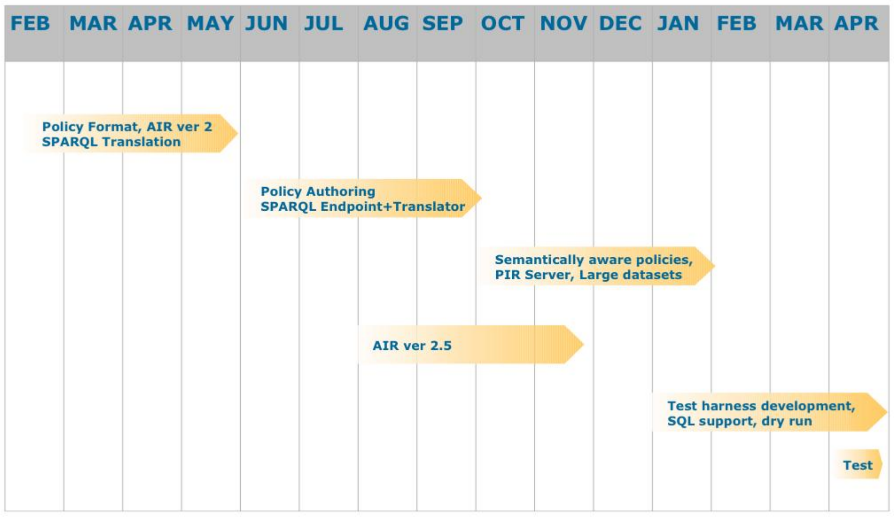


Figure 10. Timeline of Accomplishments

6. References

1. T. Berners-Lee. Primer: Getting into RDF and Semantic Web using N3. http://www.w3.org/2000/10/swap/Primer, 2005.

2. T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Prud'ommeaux, and mc schraefel. Tabulator Redux: Browsing and Writing Linked Data . In Linked Data on the Web Workshop at WWW08, 2008.

3. T. Breaux and A. Anton. Analyzing regulatory rules for privacy and security requirements. IEEE Transactions on Software Engineering, 34:5–20, 2007.

4. K. Krasnow Waterman. Pre-processing legal text: Policy parsing and isomorphic intermediate rep- resentation. In Intelligent Information Privacy Management Symposium at the AAAI Spring Symposium, 2010.

5. L. Kagal, C. Hanson, and D. Weitzner. Using dependency tracking to provide explanations for policy management. In IEEE Policy 2008, 2008.

6. L. Kagal, I. Jacobi, and A. Khandelwal. Gasping for air: Why we need linked rules and justifications on the semantic web. In Under review at the International Semantic Web Conference (ISWC), 2010.

7. A. Khandelwal, J. Bao, I. Jacobi, L. Ding, and L. Kagal. Analyzing the air language : A semantic web rule language. Technical report, Dept. of Computer Science, Rensselaer Polytechnic Institute, 2010.

8. A. Khandelwal, J. Bao, L. Kagal, I. Jacobi, L. Ding, and J. Hendler. Analyzing the air language: A semantic web (production) rules language. In The Fourth International Conference on Web Reasoning and Rule Systems (RR 2010), September 2010.

9. D. L. McGuinness, L. Ding, P. P. da Silva, and C. Chang. Pml 2: A modular explanation interlingua. In AAAI 2007 Workshop on Explanation-aware Computing, 2007.

10. T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. pages 193–216, 1988.

11. D. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. Sussman. Information accountability. pages 82–87, June 2008.

12 Pinheiro da Silva, P.; McGuinness, D. L.; and Fikes, R. 2006. A Proof Markup Language for Semantic Web Services. Information Systems 31(4–5): 381–395.

13 Deborah L. McGuinness, Li Ding, Paulo Pinheiro da Silva, Cynthia Chang. PML 2: A Modular Explanation Interlingua. AAAI 2007 Workshop on Explanation-aware Computing, Vancouver, British Columbia, Canada, July 2007.

7. List of Acronyms

PIR	Private Information Retrieval
FBI	Federal Bureau of Investigation
WWW	World Wide Web
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
OWL	Web Ontology Language
SSN	Social Security Number
AIR	Accountability In RDF
BGP	Basic Graph Pattern
URI	Uniform Resource Identifier
HR	Human Resource
PML	Proof Markup Language
SPARQL	Is the Query Language for Semantic Web Data
MIT	Massachusetts Institute of Technology
MIT-LL	MIT Lincoln Labs
UI	User Interface
FOAF	Friend of a Friend

Appendix A: AIR Rule Ontology

```
@prefix air: <http://dig.csail.mit.edu/TAMI/2007/amord/air#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<> rdfs:comment "Definition of the AIR (AMORD in RDF) policy language
version 2.0".

air:RuleSet a rdfs:Class;
   rdfs:label "A collection of rules".

air:rule a owl:ObjectProperty;
   rdfs:comment "activate a rule (belief, hidden, or ellipsed).";
   rdfs:label "Rule (belief, hidden, or ellipsed)";
   rdfs:domain air:RuleSet;
   rdfs:range air:Rule;
   rdfs:domain  [ a owl:Class;
          owl:unionOf  (
                    air:RuleSet
                    air:RuleContainer ) ].


air:Rule a rdfs:Class;
   rdfs:label "Generalized class of rules";
   rdfs:subClassOf
   [ a owl:Restriction;
      owl:minCardinality "1"^^xsd:nonNegativeInteger;
      owl:onProperty air:if ],
   [ a owl:Restriction;
      owl:maxCardinality "1"^^xsd:nonNegativeInteger;
      owl:onProperty air:description ],
   [ a owl:Restriction;
      owl:minCardinality "1"^^xsd:nonNegativeInteger;
      owl:onProperty air:action ].

air:if a owl:ObjectProperty;
   rdfs:comment "pattern to be matched against a universe.";
   rdfs:label "Pattern or condition";
   rdfs:domain air:Rule;
   rdfs:range air:Pattern.

air:action a owl:ObjectProperty;
   rdfs:comment "action (rule or assertion) to be taken.";
   rdfs:label "Rule or assertion";
   rdfs:domain air:Rule;
   rdfs:range air:AbstractContainer.
```

```
air:then a owl:ObjectProperty;
   rdfs:subPropertyOf air:action;
   rdfs:comment "action (rule or assertion) to be taken when pattern
   matches.".

air:else a owl:ObjectProperty;
   rdfs:subPropertyOf air:action;
   rdfs:comment "action (rule or assertion) to be taken when pattern
   does not match.".

air:description a owl:DataTypeProperty;
   rdfs:comment "list of strings and variables used in justification.";
   rdfs:label "Description";
   rdfs:domain  [ a owl:Class;
                owl:unionOf  (
                    air:Rule
                    air:AbstractContainer ) ] .


air:BeliefRule rdfs:subClassOf air:Abstract-rule;
   rdfs:label "Belief rule";
   rdfs:comment "rules in this class match beliefs.".

air:HiddenRule rdfs:subClassOf air:Abstract-rule;
   rdfs:label "Hidden rule";
   rdfs:comment "similar to Belief rules but its name, description,
   premises and those of its nested rules do not appear in the
   justification.".

air:ElidedRule rdfs:subClassOf air:Abstract-rule;
   rdfs:label "Elided rule";
   rdfs:comment "similar to Belief rules but in the justification
   only its flow control info appears and not its name, premises, or
   description.".


air:AbstractContainer a rdfs:Class;
   rdfs:label "Abstract class of containers for rules and assertions".


air:RuleContainer rdfs:subClassOf air:Abstract-container;
   rdfs:label "Rule collection";
   rdfs:subClassOf
   [ a owl:Restriction;
      owl:maxCardinality "1"^^xsd:nonNegativeInteger;
      owl:onProperty air:rule ].

air:AssertionContainer rdfs:subClassOf air:Abstract-container;
   rdfs:label "Assertion collection";
   rdfs:subClassOf
   [ a owl:Restriction;
      owl:maxCardinality "1"^^xsd:nonNegativeInteger;
```

```
        owl:onProperty air:assert ].

air:assert a rdf:Property;
    rdfs:label "Statement being asserted";
    rdfs:domain air:AssertionContainer;
    rdfs:range air:N3Formula;
    rdfs:comment "A graph to be asserted.".


air:N3Formula a rdfs:Class;
    rdfs:label "N3 Formula";
    rdfs:comment "this the N3 Formula class.".

air:Pattern rdfs:subClassOf air:N3Formula;
    rdfs:label "Pattern";
    rdfs:comment "pattern graph for rules.".
```

Appendix B: AIR Justification Ontology

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix pmlp: <http://inference-web.org/2.0/pml-provenance.owl#> .
@prefix pmlj: <http://inference-web.org/2.0/pml-justification.owl#> .
@prefix pmll: <http://inference-web.org/2.0/pml-lite.owl#> .
@prefix air: <http://dig.csail.mit.edu/2009/AIR/air#> .
@prefix airj: <http://dig.csail.mit.edu/2009/AIR/airjustification#> .
@prefix : <http://dig.csail.mit.edu/2009/AIR/airjustification#> .

airj:BuiltinAssertions a owl:Class;
          rdfs:label "BuiltinAssertions Event" ;
          rdfs:comment "Abstract event that asserts all assertions for a
      builtin. The asserted graph can be unbound" ;
          rdfs:subClassOf pmll:Event .

airj:ClosingTheWorld a owl:Class ;
          rdfs:label "ClosingTheWorld Event" ;
          rdfs:comment "World is closed and unmatched conditions of all
      active rules is assumed false after this." ;
          rdfs:subClassOf pmll:Event .

airj:ClosureComputation a owl:Class ;
          rdfs:label "ClosureComputation Event" ;
          rdfs:comment "This is representative of the functionality of
      the AIR reasoner" ;
          rdfs:subClassOf pmll:Event .

airj:Dereference a owl:Class ;
          rdfs:label "Dereference Event" ;
          rdfs:comment "This is notionally similar to log:semantics
      builtin which gets the graph in an N3 document" ;
          rdfs:subClassOf pmll:Event .

airj:Extraction a owl:Class ;
          rdfs:label "Extraction Event" ;
          rdfs:comment "Used to capture the operation of getting a subgraph" ;
          rdfs:subClassOf pmll:Event .

airj:BuiltinExtraction a owl:Class ;
          rdfs:label "BuiltinExtraction Event" ;
          rdfs:comment "Special extraction event to get subgraph from
      potentially unbound graph of all the builtin assertions, for
      given builtin" ;
          rdfs:subClassOf airj:Extraction .

airj:RuleApplication a owl:Class ;
```

```
            rdfs:label "RuleApplication Event" ;
            rdfs:comment "Event to describe firing of a rule" ;
            rdfs:subClassOf pmll:Event .

air:Rule rdfs:subClassOf pmll:Operation .

air:rule rdfs:subPropertyOf pmll:operation .

airj:builtin a owl:ObjectProperty ;
            rdfs:label "builtin" ;
            rdfs:subPropertyOf pmll:operation ;
            rdfs:domain airj:BuiltinAssertions .

airj:dataDependency a owl:ObjectProperty ;
            rdfs:label "data dependency" ;
            rdfs:comment "to relate two events e1 and e2 where outputdata
        of e1 is used by e2" ;
            a owl:TransitiveProperty ;
            rdfs:subPropertyOf pmll:antecedent .

airj:flowDependency a owl:ObjectProperty ;
            rdfs:label "flow dependency" ;
   rdfs:comment "to relate two events e1 and e2 where e1 must precede e2" ;
            a owl:TransitiveProperty ;
            rdfs:subPropertyOf pmll:antecedent .

airj:nestedDependency a owl:ObjectProperty ;
            rdfs:label "nested dependency" ;
            rdfs:comment "special flow dependency between RuleApplication
        events r1 and r2 where firing of rule in r1 activates rule
        fired in r2" ;
            #nested dependency is not transitive.
            a owl:FunctionalProperty ;
            rdfs:subPropertyOf airj:flowDependency ;
            rdfs:domain airj:RuleApplication ;
            rdfs:range airj:RuleApplication .

airj:inputVariableMappingList a rdf:Property ;
            rdfs:label "input variable mapping list" ;
            a owl:FunctionalProperty ;
            rdfs:domain airj:RuleApplication ;
            rdfs:subPropertyOf pmll:variableMappingList .

airj:outputVariableMappingList a rdf:Property ;
            rdfs:label "output variable mapping list" ;
            a owl:FunctionalProperty ;
            rdfs:domain airj:RuleApplication ;
            rdfs:subPropertyOf pmll:variableMappingList .

airj:outputVariableMappingListOf a rdf:Property ;
            rdfs:comment "This is notional, supposed to be inverse of
        airj:outputVariableMappingList. Strictly speaking it can not
        be a rdf:Property because it's domain is a list, but can be
        represented in N3" .

airj:branch a rdf:Property ;
             rdfs:label "branch" ;
             rdfs:comment "used to specify which actions - then-actions or
        else-actions- were fired" ;
             rdfs:domain :RuleApplication ;
             rdfs:range air:action . #air:action is a rdf:Property
```

Appendix C: DHS Report

Accountable Information Usage in Fusion Center Information Sharing Environments

Decentralized Information Group, CSAIL, Massachusetts Institute of Technology

8/30/2010

Table of Contents

List of Figures

C-1 Summary

Since 2004, when mandates were issued for an "Information Sharing Environment" [1] and a common identification standard for related individuals [2], the government and its service providers have struggled to find a mechanism that would permit the desired sharing without abandoning the rules established by law and policy. Organizations that wish to share information often hold back because they are unsure of the permissibility of the act; they find the complexity of rules and terms from multiple jurisdictions too fraught with potential for error or conflict. Organizations often do not agree to share information because they have no means to ensure that recipients would limit its use to that granted by the sender.

The Decentralized Information Group (DIG)[5] at Massachusetts Institute of Technology (MIT) explores technical, institutional, and public policy questions necessary to advance the development of global, decentralized information environments. Its prototype and concept of "accountable systems" is meant to address these issues. Accountable systems are ones which can track the usage of information within an organization. They can reason over complex policies for information handling, apply those policies to data transactions, and make determinations of compliance or non-compliance for each data sharing event. Such systems could be used for access control, privilege management, audit, aggregate reporting, redress, and risk modeling. In addition, state-of-the-art systems can explain reasoning which led to the compliance or non-compliance determination.

Fusion Centers are critical venues for sharing information between and among federal, state and local law enforcement and intelligence agencies. To enable seamless information flow while assuring that privacy and security policies are complied with, Fusion Centers require both tools that enable machine-assistant analysis of relevant policies, and mechanisms built into the information infrastructure that provide for accountable information flow.

Prior to this endeavor, the MIT team had conducted basic research into the expression of policies and the automated compliance checking of these policies in various domains using Semantic Web standards and technologies [3, 4, 5, 6]. The team leveraged its research to design and implement a set of proof-of-concept accountable information sharing tools that address needs identified in Fusion Center information sharing scenarios.

The MIT prototype is designed to respect the existing state of law, politics, and management. For example, it has the flexibility to allow each organization to create its own machine-readable version of a statute or to use one from a common library; it also assumes that organizations will overlay statutory requirements with internal legal counsel opinions or policies. It assumes that each organization will retain control of its own data and user profiles, unless that organization prefers otherwise (e.g., a small agency that wishes to avail itself of a service provider), and that policies can remain associated with data after the data is transferred.

[5] DIG is part of the Computer Science and Artificial Intelligence Lab at MIT.

In the prototype, the MIT team modeled users, each working from their own organization's systems. In our scenario, the fictitious Massachusetts analyst asks a Massachusetts system to send her memo to someone at DHS and someone at the Maryland State Police. The system determines if the criminal history information in the memo is subject to privacy protection and whether it can be released to each of those individuals under the relevant Massachusetts law. The same is true for each of the responders: each on their own system, each according to the law regulating his organization and information. In addition, the MIT prototype has enhanced capabilities which allow the system to handle more complex issues such as (1) if the term "Criminal Justice Agency" as used in one jurisdiction applies to a person in another jurisdiction that uses the same term and (2) what someone else's rules will permit them to do after you've given them your data. The prototype has received positive feedback from Fusion Center and Intelligence Community staff to whom it has been demonstrated.

The prototype uses an open standard called RDF (Resource Description Framework) for expressing all the relevant information – user profiles, metadata, and rules – and future work would create interchanges with other common standards. Additional important work to be done includes enhancing the reasoning engine to drive towards the millisecond response time needed for environments with multi-million daily transactions, extending the prototype to show how the analysis can be stored or recreated to support periodic statistical reporting or risk modeling possible changes to rules (e.g., possible changes to law, court decisions, or negotiated agreements), and researching the best methodologies for making policy decisions with incomplete or flawed information. As this work provides significant insight into the final machine-readable form into which policy must be translated, it also supports promising future work on automated policy parsing.

C-2 Methods, Assumptions, and Procedures

DIG's long-term research goal for "Accountable Systems" is to identify how systems can determine whether each use of data: (1) is/was permitted; (2) by the relevant rules for the particular data, party, and circumstance; and (3) make that decision available to access control, audit, and other technology (4) for real-time enforcement, retrospective reporting, redress, and risk modeling. This project included research into questions about how to generate the correct representation of the rule, how to attach the rule to the environment, how to reach or infer the correct information about the data, party, and circumstance, and how to pass the decision as a machine-readable binary or express it in human-readable form. In addition, we explored the question of real-time enforcement and opened the door to questions of risk modeling. It did not address issues such as efficiency, optimization, or data quality.
The research was predicated on the following assumptions:

- Web-based

The research assumes that all users are in a web-based environment. It makes no assumptions about whether the environment is a closed intranet or the open Web. Web-based systems can meet the need for obtaining decision-relevant data from beyond the locus of the transaction.

- Semantic Web

The research explores the ability of Semantic Web technologies to answer the challenges posed. Semantic web based systems can provide the level of granularity and inference that context-laden, selective privacy rules require. Semantic web based systems provide greater interoperability, reusability, and extensibility

- Authentication

This research project did not address issues of user authentication. (It deals with issues relating to user attributes and authorization.)

- Security

The research did not explicitly address security rules. Security rules, like any other data handling rules, express what is permissible given a certain set of facts or conditions. While the technology studied in this work could be applied to security rules, it was not. [6]

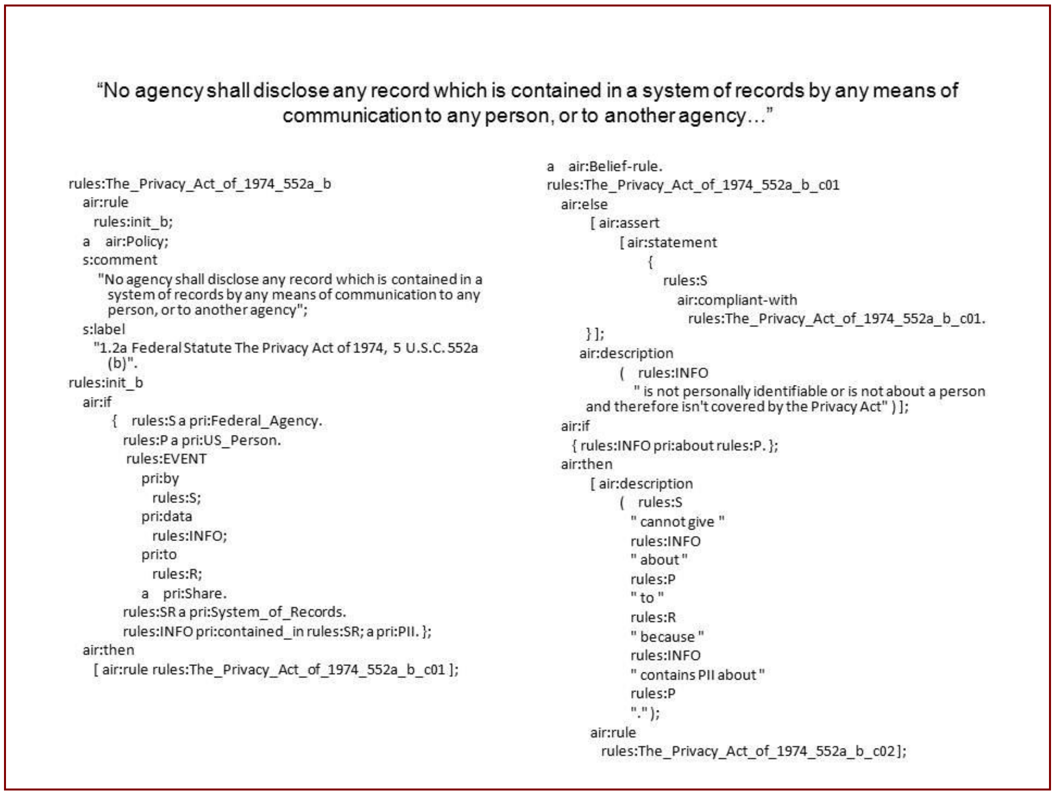
- Enhancing Accountability & Transparency

The research did not attempt to replace the full range of nuanced reasoning of which lawyers are capable. It sought to understand whether a significantly enhanced level of complex problem-solving could improve accountability and transparency for data handling rule compliance.
The primary motivation of the Semantic Web is that by associating metadata with data on the web, it enables computers to do more valuable computations than if computers did not know about the semantics of the data at hand. In particular, websites today are designed primarily for user consumption, in that machines have a hard time understanding the semantic content on any given page. If the pages also provide machine-readable metadata, automated agents can more easily perform tasks on behalf of the user.

Linked Data is the notion that by associating a unique identifier (a URI) with each piece of data in question, it is possible to create unambiguous references between pieces of data. This ability to create relationships between disparate datasets greatly increases the utility of the data, and allows computers to reason over the relationships between data. In addition, it is no longer necessary to warehouse data in one centralized location, as data in one database can refer to data in another database by URI just as easily as it can refer to data in the same database. More details about Linked Data can be found in the work of Bizer et al [7].

[6] For example, if the research modeled user(x) asking, “Can I send data(y) to person(z)?” it modeled retrieving information about person(z) from his organization’s systems necessary to answer that question. Although possible, It did not model that organization(z)’s systems asking “Is the message traffic from organization(x) permitted?” or “Is the release of person(z)’s data permitted?”

There has been much existing work in developing the technologies that enable the Semantic Web. Resource Description Framework (RDF) [8] is a model of data that provides a way to describe the relationship between resources. RDF allows for the expression of triples in the form of a subject, a predicate, and an object. Once every resource the system wants to reason over (actors, documents, transactions, policies, etc.) has been associated with a URI, it is possible to use RDF to describe the relationship between these resources (e.g. a subject "transaction", a predicate "compliant with", and an object "Federal Privacy Act"). In addition to providing a way to talk about the relationships between data, the system also needs a way to describe the hierarchy of objects and how they relate to each other. The system accomplishes this through the Web Ontology Language (OWL) [9].



"No agency shall disclose any record which is contained in a system of records by any means of communication to any person, or to another agency…"

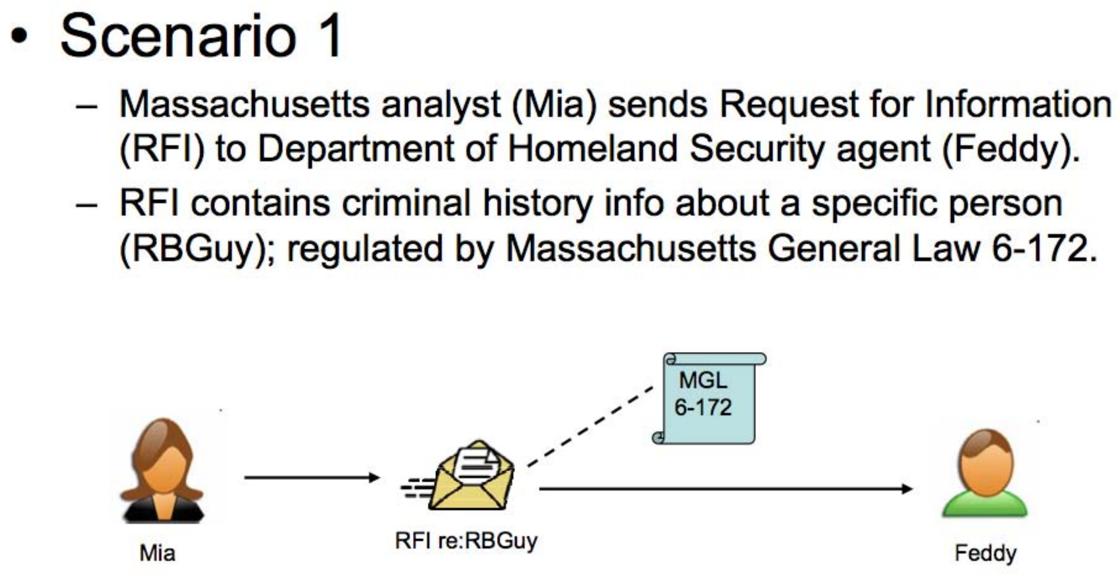
```
rules:The_Privacy_Act_of_1974_552a_b
  air:rule
    rules:init_b;
  a   air:Policy;
  s:comment
    "No agency shall disclose any record which is contained in a
      system of records by any means of communication to any
      person, or to another agency";
  s:label
    "1.2a Federal Statute The Privacy Act of 1974, 5 U.S.C. 552a
      (b)".
rules:init_b
  air:if
       {   rules:S a pri:Federal_Agency.
         rules:P a pri:US_Person.
          rules:EVENT
            pri:by
              rules:S;
            pri:data
              rules:INFO;
            pri:to
              rules:R;
            a   pri:Share.
         rules:SR a pri:System_of_Records.
         rules:INFO pri:contained_in rules:SR; a pri:PII. };
  air:then
    [ air:rule rules:The_Privacy_Act_of_1974_552a_b_c01 ];
```

```
a   air:Belief-rule.
rules:The_Privacy_Act_of_1974_552a_b_c01
  air:else
       [ air:assert
            [ air:statement
                 {
                    rules:S
                       air:compliant-with
                         rules:The_Privacy_Act_of_1974_552a_b_c01.
     } ];
    air:description
            (   rules:INFO
               " is not personally identifiable or is not about a person
     and therefore isn't covered by the Privacy Act" ) ];
  air:if
    { rules:INFO pri:about rules:P. };
  air:then
       [ air:description
            (   rules:S
              " cannot give "
              rules:INFO
              " about "
              rules:P
              " to "
              rules:R
              " because "
              rules:INFO
              " contains PII about "
              rules:P
              "." );
       air:rule
         rules:The_Privacy_Act_of_1974_552a_b_c02 ];
```

Figure C-11: Representing policy in AIR.

The AIR (Accountability in RDF) policy language, as described by Kagal et al [10], allows for the expression of policies as a series of patterns representing criteria to be met for compliance with a particular rule. This seems to fit well with legal rules which often are referred to as having "elements", such as the five fair uses of copyright. The system uses a forward chaining reasoner, cwm [11], to evaluate an information transaction against a relevant policy. The reasoner has incorporated a Truth Maintenance System [12], a dependency tracking mechanism, which allows the system to retain the dependencies upon which it relied to form its conclusions.

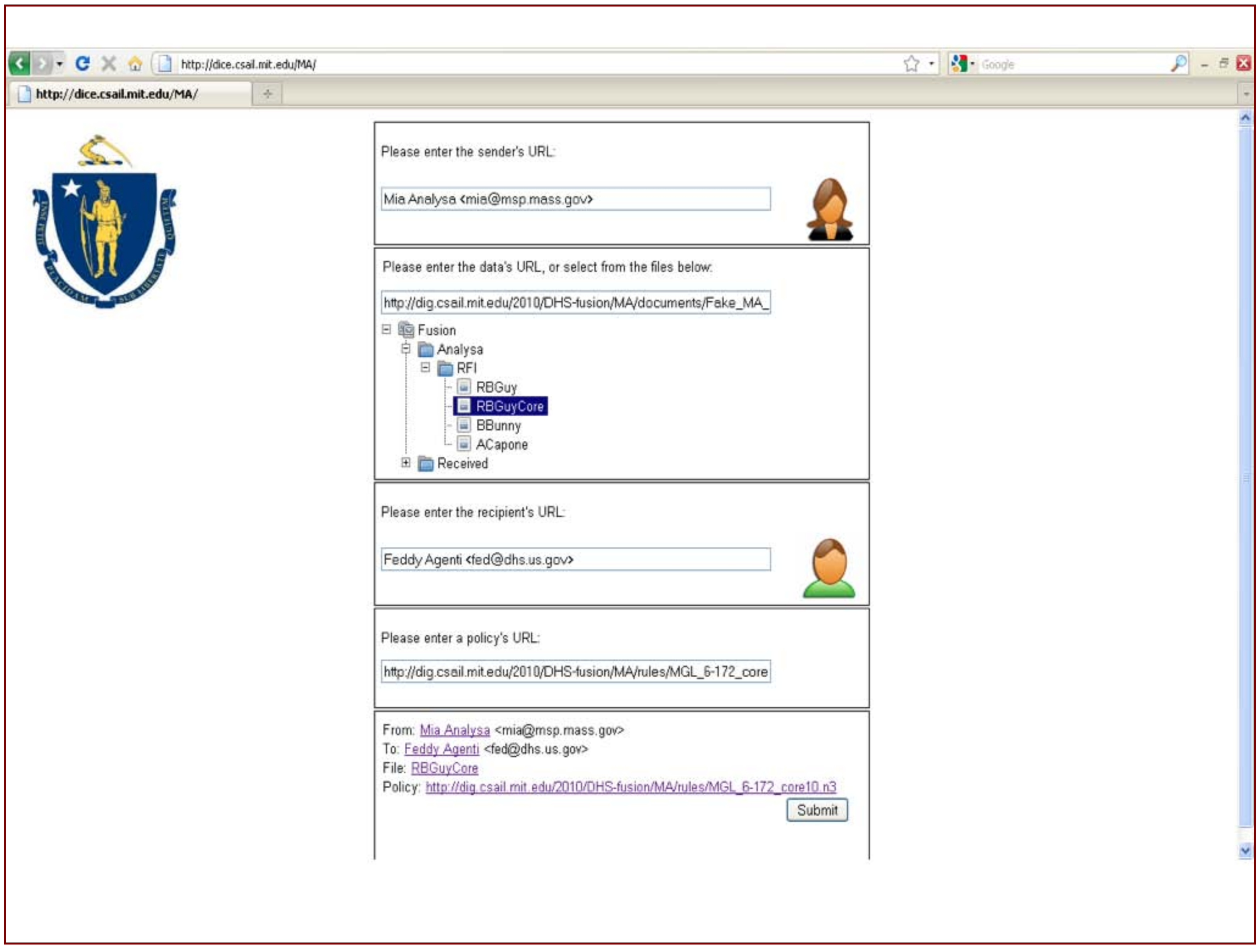
Fusion Center personnel were interviewed regarding current information sharing and the methodology for ensuring rules compliance. Target jurisdictions were identified and then laws in those jurisdictions protecting personal information. From among those, three complex rules were selected to be modeled and tested in the prototype. Fictitious scenarios were written and the scenarios were vetted by Fusion Center analysts as reasonable models of the sorts of communications that occur.



- Scenario 1
  - Massachusetts analyst (Mia) sends Request for Information (RFI) to Department of Homeland Security agent (Feddy).
  - RFI contains criminal history info about a specific person (RBGuy); regulated by Massachusetts General Law 6-172.

Figure C-2: Simple expression of scenario.

The component pieces were then created. The Federal Privacy Act [13], in its entirety, and large segments of a Maryland and a Massachusetts law [14, 15] were represented in AIR. This required an iterative process and consultation between those producing the code and a lawyer. Mock user profiles were created, initially representing an analyst in Massachusetts, a detective in Maryland, and an Immigration and Customs Enforcement (ICE) agent at the Department of Homeland Security (DHS). The user profiles were written using RDF, and adhered to the ontology from the Friend-of-a-Friend (FOAF) [17] design. FOAF allows for greater flexibility of information than the typical system administration profile, because each organization can add additional relevant pieces of information to any profile without having to change some central schema. Mock documents were created, representing the Massachusetts analyst's request for information (RFI) about a possible criminal suspect with a previous criminal Massachusetts criminal history, the DHS response indicating a match and requesting personal contact, and a Maryland response with additional Maryland criminal history. The documents were created as PDF files and annotated with RDF using existing features in Adobe Acrobat.



Please enter the sender's URL:

Mia Analysa <mia@msp.mass.gov>

Please enter the data's URL, or select from the files below:

http://dig.csail.mit.edu/2010/DHS-fusion/MA/documents/Fake_MA_

- Fusion
  - Analysa
    - RFI
      - RBGuy
      - RBGuyCore
      - BBunny
      - ACapone
  - Received

Please enter the recipient's URL:

Feddy Agenti <fed@dhs.us.gov>

Please enter a policy's URL:

http://dig.csail.mit.edu/2010/DHS-fusion/MA/rules/MGL_6-172_core

From: Mia Analysa <mia@msp.mass.gov>
To: Feddy Agenti <fed@dhs.us.gov>
File: RBGuyCore
Policy: http://dig.csail.mit.edu/2010/DHS-fusion/MA/rules/MGL_6-172_core10.n3

Figure C-3: Transaction Simulator

A user interface was created to allow clear visualization of data transaction activities. In the "Transaction Simulator" the user can explicitly express that he wants to send a particular document to a particular person, as he might via email or a web portal. Three "Transaction Simulator" UIs were created, one each for Massachusetts, Maryland, and DHS. Code was written so that pressing "submit" would cause metadata to be pulled from the people's profiles, the document, and rules and submitted to the reasoner. Output from the reasoner is then fed to "Tabulator" [17], a Semantic Web browser with multiple viewing panes.

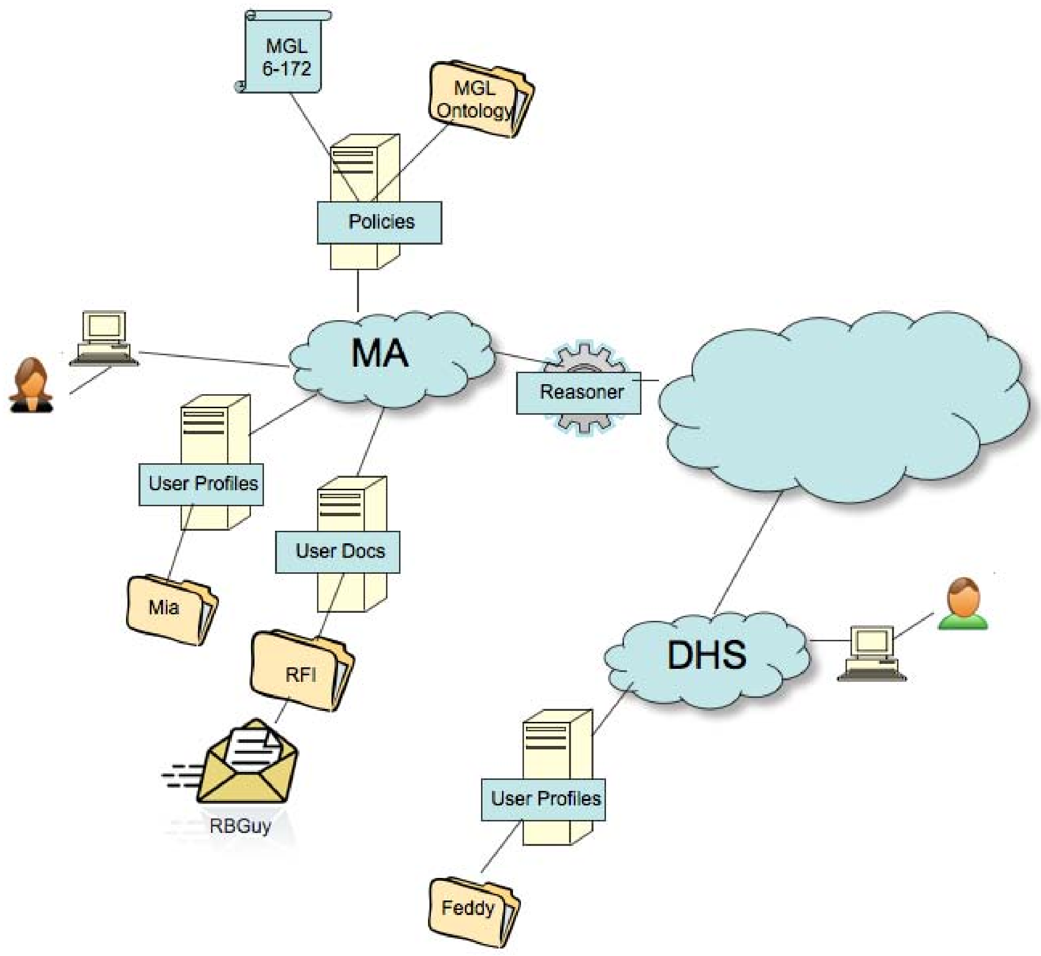


Figure C-4: Sample feeds to reasoner

C-3 Results and Discussions

DIG modeled and executed six scenarios through the reasoner, which was sufficiently powerful enough to produce justifications for each of the designed scenarios. In addition, the system has sufficient capability and flexibility such that it is possible to run new scenarios, mixing and matching the component pieces in unplanned ways. From the research perspective, this exercise demonstrated the first order increase in scalability. In earlier work [18], the MIT group fed to the reasoner only the input necessary to reach a correct conclusion. In this work, it fed the reasoner a significant number of rule patterns and facts that were unnecessary to the conclusion. For example, a “simple” transaction involved 27 facts about the sender, 25 about the recipient, 6 about the document, and 35 sub-rules; there is no inherent limitation on the number of items that will be relevant and used from each category, creating the potential for more than 100,000 pattern match combinations. The MIT group confirmed, so long as the rules are expressed correctly, that the correct result will be produced – only the appropriate sub-rules will be found to support relevant beliefs, and only the relevant facts will be reported as dependencies. As the “so long as” clause implies, the work showed the importance and necessity of validation, i.e. the ability to determine that the rules have been expressed correctly in their entirety –both the pattern and its relationship to all other patterns (e.g., conditions, exceptions, order). The MIT team also proved that “broken” or undefined bits did not necessarily keep the system from reaching a conclusion. For example, a scenario was run under the Massachusetts criminal records release law and the proposed recipient had a malformed tag which was intended to identify him as a member of a criminal justice agency but failed to do so, the system correctly bypassed the rules applying to members of criminal justice agencies and found a later sub-rule in the policy applicable, finding him entitled to such information as any member of the public may receive.

For the prototype to be accessible for evaluation and validation by a broad array of interested parties (e.g., government executives, policy leaders, lawyers, and the professionals who need to share the information), the sub-rules are coded in the order in which they appear in statute and annotated with their legal citations. This is particularly challenging because law is generated through negotiation and does not generally follow formal logic structures; programmers are trained to correct that flaw and reduce the rules to their most efficient form. By representing the rules as written, it is possible to review them in Tabulator’s data pane as a graphical view of consecutive series of if-then-else statements tied to the sub-rule citations.

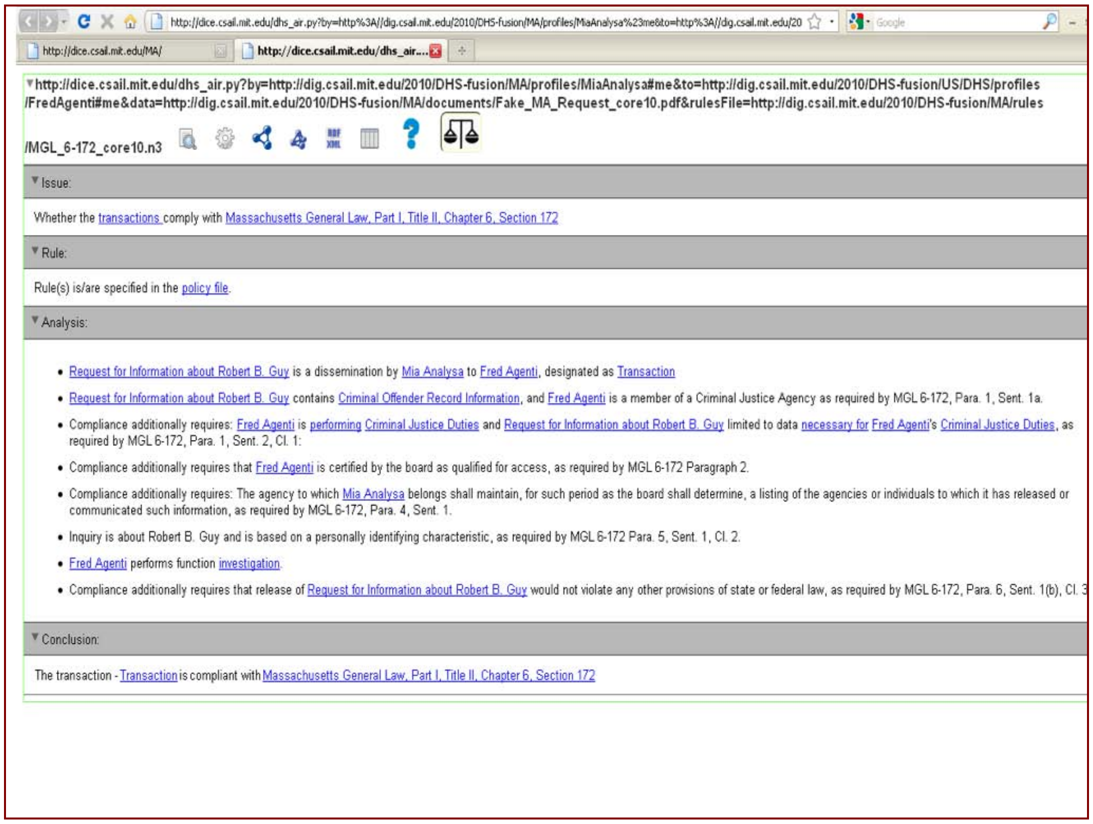
A particularly important challenge for the applications that were explored is that the data is being sent between organizations with different personnel and different information systems. If the information about users, data, and policies which already exists in different languages or platforms can all be annotated using a common standard, it is not necessary to transform the information to a common language or platform, nor to move it to a single repository to reason over it. In the MIT prototype, the data was annotated with RDF; by assigning a URI to each resource to be talked about it was possible for each organization in the simulation to keep their data independent from each other. Systems located at each organization are able to dereference data on other organizations’ systems, and reason over data and personnel from those organizations. This decentralized design does not require a central agency to watch over all transactions to ensure compliance with policy; it is possible for each organization to ensure that the transactions that they engage in are compliant with the policies that are relevant to the people, organization, data, and/or circumstance.

In addition, since there is a way for organizations to describe the way they store data and the policies that are relevant to them, it was possible to describe and address the nuances of each organization and their data in the data itself. OWL was used to specify the terms that each organization uses through an ontology. It also permitted each jurisdiction to independently represent relationships between entities (e.g. a” police officer” is “sworn law enforcement”). Using OWL permitted reasoning between the objects in two different organizations without implicitly assuming that organizations agree on the terminology being used. For example, the prototype won’t assume that a Maryland “police officer” is interchangeable with a Massachusetts “police officer” unless that relationship is made explicit.

Because of the design choice of a forward chaining reasoner, the reasoner itself could not issue calls for more information. Pre-processing was required to deliver all the necessary data to the reasoner. For example, the prototype automatically identifies to the reasoner the URL for the sender's profile, the proposed recipient's profile, and the target data; it also pre-processes by crawling those files for references to other policies or ontologies and delivers those URLs to the reasoner as well. In addition, functionality was built so that the system searches rules for any assertions that it will need (e.g., where a rule calls for a subjective judgment), queries the user, and delivers the result to the reasoner.

The design choice of a TMS was extremely useful because it allows users to see the basis for a decision, a function not available from some other policy reasoner's. The output, viewed in Tabulator, provides a quasi-grammatical statement of whether the transaction is compliant or non-compliant and additional statements explaining exactly which facts where applied to which requirements in a rule, including the legal citation for the sub-rule. These near-sentences are compiled by the prototype based upon the available dependency tracking information stored by the TMS in the process of making its decision.

As part of this research, the prototype was demonstrated to a variety of relevant persons – ranging from Fusion Center analysts to Intelligence Community management, both technical and operational. The reactions were very positive in that such an accountable system could fulfill government obligations to ensure that information sharing is handled in a policy compliant manner and to provide a level of transparency to users. The most significant resistance received was from an analyst
supervisor who perceived this as having the potential to be a management surveillance tool to question the ability of individual analysts to know and comply with all rules; however, even that individual believed that the mechanism would be quite helpful when necessary to apply the rules of another jurisdiction (i.e., not one's own) and for use as a workflow management tool. Conversely, the analysts at a demo the next day were so enthusiastic that they wanted to know if they could build and use the FOAF-based user profiles immediately.


http://dice.csail.mit.edu/dhs_air.py?by=http%3A//dig.csail.mit.edu/2010/DHS-fusion/MA/profiles/MiaAnalysa%23me&to=http%3A//dig.csail.mit.edu/20

http://dice.csail.mit.edu/MA/

http://dice.csail.mit.edu/dhs_air....

http://dice.csail.mit.edu/dhs_air.py?by=http://dig.csail.mit.edu/2010/DHS-fusion/MA/profiles/MiaAnalysa#me&to=http://dig.csail.mit.edu/2010/DHS-fusion/US/DHS/profiles/FredAgenti#me&data=http://dig.csail.mit.edu/2010/DHS-fusion/MA/documents/Fake_MA_Request_core10.pdf&rulesFile=http://dig.csail.mit.edu/2010/DHS-fusion/MA/rules/MGL_6-172_core10.n3

Issue:

Whether the transactions comply with Massachusetts General Law, Part I, Title II, Chapter 6, Section 172

Rule:

Rule(s) is/are specified in the policy file.

Analysis:

- Request for Information about Robert B. Guy is a dissemination by Mia Analysa to Fred Agenti, designated as Transaction
- Request for Information about Robert B. Guy contains Criminal Offender Record Information, and Fred Agenti is a member of a Criminal Justice Agency as required by MGL 6-172, Para. 1, Sent. 1a.
- Compliance additionally requires: Fred Agenti is performing Criminal Justice Duties and Request for Information about Robert B. Guy limited to data necessary for Fred Agenti's Criminal Justice Duties, as required by MGL 6-172, Para. 1, Sent. 2, Cl. 1:
- Compliance additionally requires that Fred Agenti is certified by the board as qualified for access, as required by MGL 6-172 Paragraph 2.
- Compliance additionally requires: The agency to which Mia Analysa belongs shall maintain, for such period as the board shall determine, a listing of the agencies or individuals to which it has released or communicated such information, as required by MGL 6-172, Para. 4, Sent. 1.
- Inquiry is about Robert B. Guy and is based on a personally identifying characteristic, as required by MGL 6-172 Para. 5, Sent. 1, Cl. 2.
- Fred Agenti performs function investigation.
- Compliance additionally requires that release of Request for Information about Robert B. Guy would not violate any other provisions of state or federal law, as required by MGL 6-172, Para. 6, Sent. 1(b), Cl. 3

Conclusion:

The transaction - Transaction is compliant with Massachusetts General Law, Part I, Title II, Chapter 6, Section 172

Figure C-5: Sample "lawyer" pane with detailed explanation of conclusion

C-4 Conclusions

DIG succeeded in its research goal, determining that Semantic Web technologies are relevant and usable for the challenge of ensuring compliant information sharing, specifically in this case compliance with privacy-related rules. The AIR policy language was able to express a wide variety of rules requirements and language, as well as necessary relationships among people, data, circumstances, and rules. The reasoner was able to function and achieve correct results when presented with many sub-rules and facts, including those that were not relevant or functional. The demonstration was reasonably well understood by individuals with no prior knowledge of the project and identified as a potential method by which the government could fulfill its obligations for compliance and accountability.

C-5 Recommendations of Future Work

Automated reasoning over a large corpus of rules and facts presents a significant computational load. A single law, such as the Privacy Act, can contain well more than a hundred sub-rule patterns to be matched. Resolving each of these patterns can involve dozens of other rules containing yet more patterns, requiring the system to make many thousands of discrete deductions. Even a single act of information sharing can be regulated by dozens of other polices. While audit functions may be accomplished in a less time sensitive manner and state Fusion Centers are not sending large volumes of data, any system providing accountability for millions of transactions as they occur must be able to accomplish this reasoning in the moment between hitting the “send” button and the expectation that the transaction is complete. Research should be performed to identify the method to most effectively increase the speed of the reasoning.

The ability to reason perfectly over transactions is limited by the availability and quality of the data over which an accountable system seeks to reason. In the real world, laws and facts are often inconsistent. Government and industry continue to make strides in tagging data to expose more meaning and provenance and in increasing the granularity of information about users. At the same time, work has begun to improve the ability to expose more environmental and contextual information at the machine level. It is, however, unreasonable to wait until all such activities have been perfected and completed to take advantage of the capability of machines to handle large and difficult problems. Research should be performed to determine how an accountable system might be able to accept incomplete information about the elements of a data transaction, make a best judgment, even in the face of inconsistency, and explain the level of risk associated with its conclusion, much in the way that people do when faced with similar situations— as they regularly are.

An accountable system will be trusted only if it is sufficiently transparent to be validated. Such validation is required at every step of the process. Lawyers and programmers will want to know that a rule has been correctly stated in code; they must be able to see the text of the law, regulation, or policy concurrently with some representation of what is in the system; research is needed both on how to accomplish this at all and how to accomplish it to a level of human satisfaction. While the prototype provides for means to interrogate the system and ask it to justify its conclusions, research must be done to determine a method to allow them to pose hypothetical’s (without mock data and personal profiles) to more quickly validate the correctness of the rule representation and to perform risk modeling regarding possible rule changes. And, though the TMS theoretically would permit the storage of the dependencies from each decision and allow for aggregate statistical reporting (e.g., to meet government transparency and accountability requirements), it has not been determined how specifically to achieve this goal.

To date, a single data or architectural standard has not been implemented throughout the federal government. And, because the United States Constitution gives sovereignty to the states, they need not follow a federal standard for accountable systems. Therefore, research needs to be performed to validate theories regarding “rules interchange”. MIT has discussed the possibility of collaborative testing and research with parties producing machine-readable rules and metadata under other standards.

C-6 References

[1] Required first by the White House under Executive Order 13356 (Aug. 27, 2004) and, later, by Congress under Section 1016 of the Intelligence Reform and Terrorism Prevention Act (Dec. 17, 2004).

[2] Required by Homeland Security Presidential Directive 12 (Aug. 27, 2004).

[3] Daniel Weitzner, Hal Abelson, Tim Berners-Lee, Joan Feigenbaum, Jim Hendler, Gerry Sussman, Information Accountability, *Communications of the ACM,* June 2008.

[4] Lalana Kagal, Chris Hanson, and Daniel Weitzner, "Integrated Policy Explanations via Dependency Tracking", IEEE Policy 2008. http://dig.csail.mit.edu/2008/Papers/IEEE%20Policy/air-overview.pdf

[5] L. Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner, "Using Semantic Web Technologies for Open Policy Management on the Web", 21st National Conference on Artificial Intelligence (AAAI 2006), July 2006. http://dig.csail.mit.edu/2006/Papers/AAAI/

[6] Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler, "N3Logic: A Logical Framework for the World Wide Web", Journal of Theory and Practice of Logic Programming (TPLP), Special Issue on Logic Programming and the Web, 2007. http://arxiv.org/abs/0711.1533

[7] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," International Journal on Semantic Web and Information Systems, 2009.

[8] O. Lassila and R. Swick, "Resource description framework (RDF) model and syntax," World Wide Web Consortium. http://www.w3.org/TR/WDrdf-syntax

[9] D. McGuinness, F. Van Harmelen et al., "OWL web ontology language overview," W3C recommendation, vol. 10, pp. 2004–03, 2004.

[10] L. Kagal, C. Hanson, and D. Weitzner, "Using dependency tracking to provide explanations for policy management," in IEEE Policy, vol. 2008. Citeseer, 2008.

[11] T. Berners-Lee et al., "Cwm: A general purpose data processor for the semantic web," Project Web site, W3C, 2006.

[12] J. Doyle, "A truth maintenance system* 1," Artificial intelligence, vol. 12, no. 3, pp. 231–272, 1979.

[13] 5 USC § 552a.

[14] MCCL, Ch. 12, Subtitle 15 § 01.11.

[15] MGL, Ch.6 § 172.

[16] D. Brickley and L. Miller, “FOAF vocabulary specification 0.91,” Namespace document, FOAF Project (November 2007). http://xmlns.com/foaf/0.1

[17] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, “Tabulator: Exploring and analyzing linked data on the semantic web,” in Proceedings of the 3rd International Semantic Web User Interaction Workshop, vol. 2006.

[18] D. Weitzner, H. Abelson, T. Berners-Lee, C. Hanson, J. Hendler, L. Kagal, D. McGuinness, G. Sussman, and K. Waterman, “Transparent accountable inferencing for privacy risk management,” in AAAI Spring Symposium on The Semantic Web meets eGovernment. AAAI Press, Stanford University, USA. Citeseer, 2006.